VirtualHub, Mode d'emploi

Table des matières

1. Introduction **1**

2. Installation **3**

2.1. Linux et USB 3

3. Configuration et test des modules **5**

3.1. Localisation des modules 6

3.2. Test des modules 6

3.3. Configuration des modules 6

3.4. Upgrades des firmwares 7

4. Utilisation du VirtualHub comme une passerelle **9**

4.1. Limitations 9

5. Contrôle d'accès **11**

5.1. Accès "admin" 11

5.2. Accès "user" 12

5.3. Influence sur les API 12

6. Interactions avec l'extérieur **13**

6.1. 13

6.2. User defined callback 14

6.3. Yocto-API callback 15

6.4. Cosm 15

7. Paramètres de la ligne de commande **17**

Index **19**

1. Introduction

Le *VirtualHub* est une application essentiellement destinée à gérer les modules USB conçus par Yoctopuce. C'est un genre de boîte à outils qui a pour but:

- D'offrir l'accès aux modules USB depuis des languages qui, tels que Javascript et PHP, ne permettent pas d'accéder couches matérielles d'un ordinateur.
- D'offrir l'accès aux modules USB à travers une connection réseau, et ce dans tous les langages disponibles.
- Configurer et tester les modules USB Yoctopuce
- D'offrir la connectivité nécessaire pour permettre aux modules Yoctopuce d'interragir avec un service cloud-based.

Le *VirtualHub* n'est **pas** indispensable pour contrôler directement des modules Yoctopuce dans les langages qui permettent un accès natif au hardware (C++, Delphi, Python, Visual Basic, C#, Android, API en ligne de commande). Dans ces langages, les modules USB Yoctopuce peuvent être contrôlés **directement** sans même l'aide d'un driver.

Le *VirtualHub* est disponible pour les systèmes d'exploitation Windows, Mac OS X et Linux (Intel et ARM). Son fonctionnement est identique sur les trois systèmes.

2. Installation

Le *VirtualHub* ne nécessite pas véritablement d'installation. C'est un simple fichier exécutable. Copiez-le où bon vous semble, et lancez-le depuis une ligne de commande. Aucun driver n'est nécessaire.

Sous Windows, si vous ne souhaitez pas devoir lancer explicitement le*VirtualHub* à chaque fois que vous en avez besoin, vous pouvez l'installer en service: il vous suffit de le lancer une fois avec l'option `-i` et le *VirtualHub* se lancera automatiquement à chaque fois que l'ordinateur démarrera.

Le *VirtualHub* a besoin de sauvegarder quelques paramètres, ces paramètres seront sauvés dans un fichier `.virtualhub.dat` qui sera placé dans le répertoire *AppData* de l'utilisateur sous Windows, et dans le *homedir* de l'utilisateur sous Linux et Mac OS X. Ce comportement peut être modifié à l'aide d'une option sur la ligne de commande.

2.1. Linux et USB

Pour fonctionner correctement sous Linux le VirtualHub a besoin d'avoir accès en écriture à tous les périphériques USB Yoctopuce. Or, par défaut, sous Linux les droits d'accès des utilisateurs non-root à USB sont limités à la lecture. Afin d'éviter de devoir lancer les exécutables en tant que root, il faut créer une nouvelle règle *udev* pour autoriser un ou plusieurs utilisateurs à accéder en écriture aux périphériques Yoctopuce.

Pour ajouter une règle *udev* à votre installation, il faut ajouter un fichier avec un nom au format "`##-nomArbitraire.rules`" dans le répertoire "`/etc/udev/rules.d`". Lors du démarrage du système, *udev* va lire tous les fichiers avec l'extension "`.rules`" de ce répertoire en respectant l'ordre alphabétique (par exemple, le fichier "`51-custom.rules`" sera interprété APRES le fichier "`50-udev-default.rules`").

Le fichier "`50-udev-default`" contient les règles *udev* par défaut du système. Pour modifier le comportement par défaut du système, il faut donc créer un fichier qui commence par un nombre plus grand que 50, qui définira un comportement plus spécifique que le défaut du système. Notez que pour ajouter une règle vous aurez besoin d'avoir un accès root sur le système.

Dans le répertoire `udev_conf` de l'archive du *VirtualHub*[1] pour Linux, vous trouverez deux exemples de règles qui vous éviterons de devoir partir de rien.

[1] http://www.yoctopuce.com/EN/virtualhub.php

Exemple 1: 51-yoctopuce.rules

Cette règle va autoriser tous les utilisateurs à accéder en lecture et en écriture aux périphériques Yoctopuce USB. Les droits d'accès pour tous les autres périphériques ne seront pas modifiés. Si ce scénario vous convient il suffit de copier le fichier "`51-yoctopuce_all.rules`" dans le répertoire "`/etc/udev/rules.d`" et de redémarrer votre système.

```
# udev rules to allow write access to all users
# for Yoctopuce USB devices
SUBSYSTEM=="usb", ATTR{idVendor}=="24e0", MODE="0666"
```

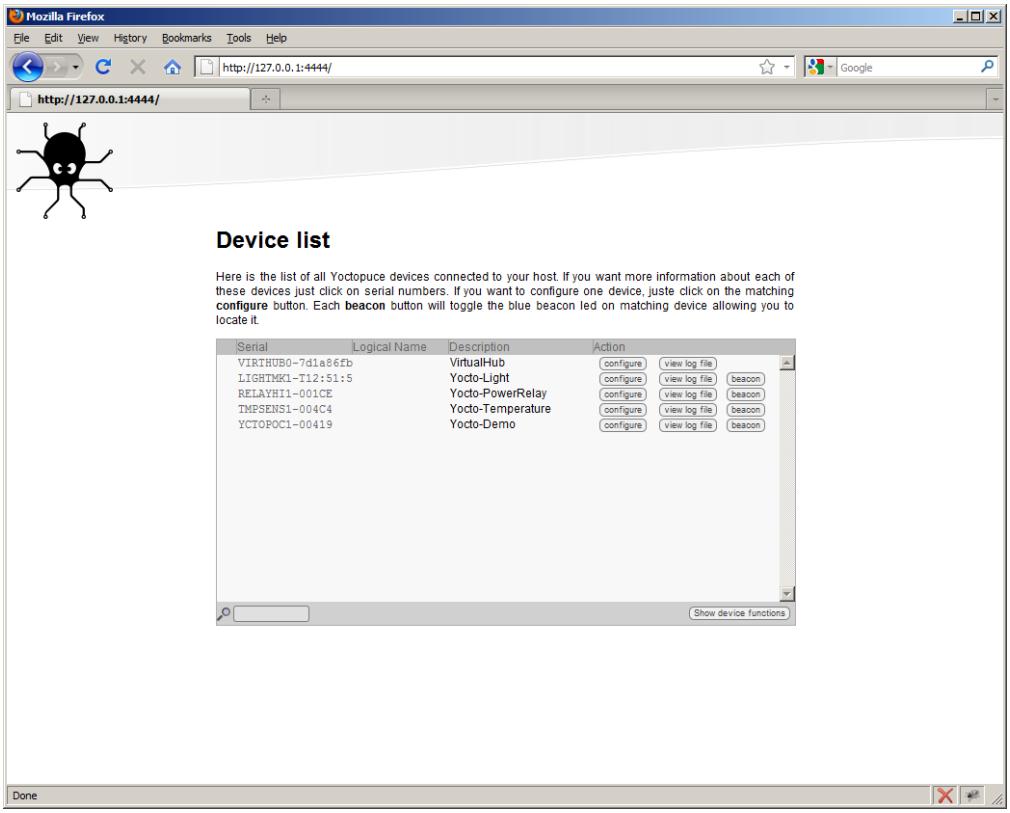
Exemple 2: 51-yoctopuce_group.rules

Cette règle va autoriser le groupe "yoctogroup" à accéder en lecture et écriture aux périphériques Yoctopuce USB. Les droits d'accès pour tous les autres périphériques ne seront pas modifiés. Si ce scénario vous convient il suffit de copier le fichier "`51-yoctopuce_group.rules`" dans le répertoire "`/etc/udev/rules.d`" et de redémarrer votre système.

```
# udev rules to allow write access to all users of "yoctogroup"
# for Yoctopuce USB devices
SUBSYSTEM=="usb", ATTR{idVendor}=="24e0", MODE="0664", GROUP="yoctogroup"
```

3. Configuration et test des modules

Le *VirtualHub* permet de tester et configurer vos modules Yoctopuce. Pour ce faire assurez-vous qu'un *VirtualHub* tourne sur la machine à laquelle sont connectés les modules, puis ouvrez votre navigateur internet favori[1]. Connectez-vous en *HTTP* au port 4444 de la machine sur laquelle tourne le *VirtualHub*. S'il s'agit de la machine locale, utilisez l'adresse `http://127.0.0.1:4444`. La liste de vos modules devrait apparaître.

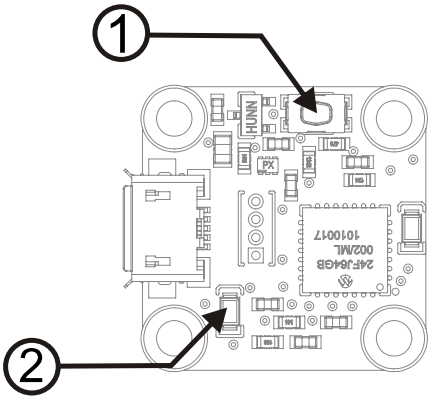


Interface Web du VirtualHub.

[1] L'interface du *VirtualHub* est régulièrement testée sur Internet Explorer 6+, Firefox 3.5+, Chrome et Safari. Elle ne fonctionne pas avec Opéra

3.1. Localisation des modules

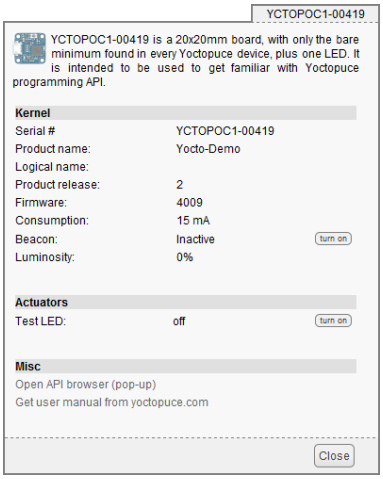
L'interface principale vous montre une ligne par module connecté, si vous avez plusieurs modules du même modèle, vous pouvez localiser un module particulier en cliquant sur le bouton **beacon** correspondant: cela aura pour effet de faire clignoter la led bleue du module et d'afficher sur l'interface une pastille bleue au début de la ligne correspondante. Vous pouvez faire la même manipulation en appuyant sur le Yocto-bouton d'un module connecté.



Yocto-bouton(1) et led(2) de localisation d'un module Yocto-Demo. Ces deux éléments sont toujours placés au même endroit, quelque soit le module.

3.2. Test des modules

Pour tester un module, cliquez simplement sur le numéro de série d'un module dans l'interface, une fenêtre spécifique au module s'ouvrira. Cette fenêtre permet généralement d'activer les fonctions principales du module. Reportez vous au manuel du module correspondant pour plus de détails.[2]

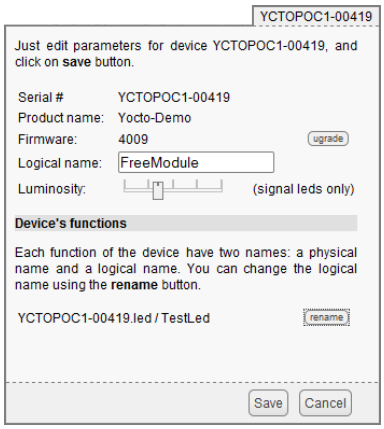


Fenêtre "détails" du module Yocto-Demo.

3.3. Configuration des modules

Vous pouvez configurer un module en cliquant sur le bouton **Configure** correspondant dans l'interface principale, une fenêtre spécifique au module s'ouvre alors. Cette fenêtre permet au minimum de donner un nom logique au module ainsi que de mettre à jour son firmware. Reportez-vous au manuel du module correspondant pour plus de détails.

[2] Vous n'êtes pas obligé d'avoir un *VirtualHub* plus récent qu'un module pour le tester/configurer: tous les éléments spécifiques aux interfaces des modules sont stockés dans la ROM des modules, et non pas dans le *VirtualHub*.



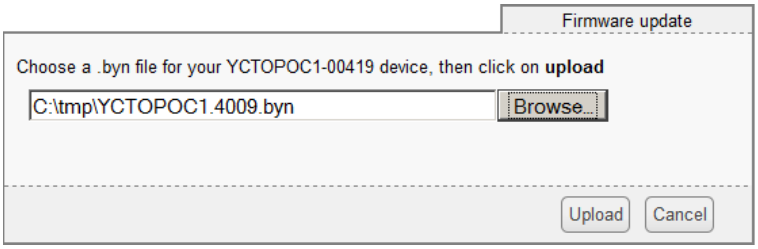
Fenêtre "configure" du module Yocto-Demo.

3.4. Upgrades des firmwares

Les modules Yoctopuce sont en fait de véritables ordinateurs, ils contiennent même un petit serveur web. Et comme tous les ordinateurs, il est possible de mettre à jour leur logiciel de contrôle (firmware). Des nouveaux firmwares pour chaque module sont régulièrement publiés, ils permettent généralement d'ajouter de nouvelles fonctionnalités au module, et/ou de corriger d'éventuels bugs[3].

Méthode recommandée

Pour mettre à jour le firmware d'un module, Vous devez d'abort vous procurer le firmware, il peut être téléchargé depuis la page produit du module sur le site de Yoctopuce [4]. L'interface propose aussi un lien direct si elle détecte que le firmware n'est pas à jour[5]. Ces firmwares se présentent sous la form de fichier `.byn` de quelques dizaines de Kilo-octets, sauvez celui qui vous intéresse sur votre disque local



Fenêtre de mise à jour du firmware.

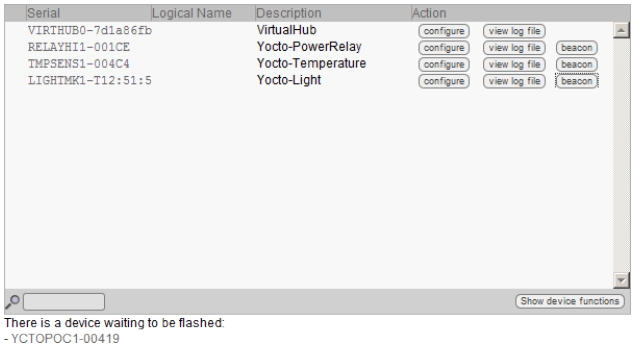
Une fois votre fichier de firmware disponible localement, ouvrez la fenêtre **configuration** d'un module et cliquez sur le bouton **upgrade**. L'interface va vous demander de choisir le fichier de firmware que vous désirez utiliser. Entrez le nom du fichier et cliquez sur **Upload**. A partir de là, tout est automatique, le *VirtualHub* va faire redémarrer le module en mode "mise à jour", mettre à jour le firmware, puis rédémarrer le module en mode normal. Les réglages de configuration du module seront préservés. Ne débranchez pas le module pendant la procédure de mise à jour.

Méthode alternative 1

Si la mise à jour d'un module se passe mal, en particulier si le module à été débranché pendant le processus, il risque fort de ne plus fonctionner et de ne plus apparaître dans la listes des modules. Dans ce cas débranchez-le, attendez quelques secondes, et rebranchez-le en maintenant le yocto-bouton appuyé. Cela a pour effet de faire démarrer le module en mode "mise à jour". Ce mode de fonctionnement est protégé contre les corruptions, et devrait toujours être accessible. Une fois le module rebranché, provoquez un rafraîchissement de la liste des modules dans l'interface du *VirtualHub* et votre module devrait être listé dans le bas de l'interface. Cliquez dessus pour mettre à

[3] Ne faites jamais confiance à des gens qui vous disent que leur logiciel n'a pas de bug :-)
[4] www.yoctopuce.com
[5] A condition qu'elle ai réussi à accéder au site web de Yoctopuce

jour son firmware. Ce mode de mise à jour est une procédure de récupération, elle ne sauvegarde pas les réglages du module.



Les modules en mode "mise à jour" sont listés dans l'interface.

Méthode alternative 2

Vous pouvez aussi mettre à jour le firmware d'un module en utilisant le *VirtualHub* en ligne de commande. Connectez le module en maintenant son yocto-bouton appuyé puis utilisez la commande-line suivante:

```
virtualhub -f numero_de_serie fichier.byn
```

Notez que cela nécessite de connaître numéro de série de votre module. Ce mode de mise à jour est une procédure de récupération, elle ne sauvegarde pas les réglages du module.

4. Utilisation du VirtualHub comme une passerelle

La fonction la moins spectaculaire, mais néanmoins la plus utile du *VirtualHub* consiste à offrir une passerelle réseau pour contrôler les modules. Cela permet d'une part d'offrir un accès aux languages comme Javascript, qui par nature interdisent d'accéder aux ressources physiques d'une machine. D'autre part cela permet d'offrir un accès aux modules à travers le réseau dans tous les languages: les libraries Yoctopuces sont en effet capables de se connecter à un *VirtualHub* à travers le réseau.

Pour utiliser le *VirtualHub* comme passerelle, il vous suffit de le lancer en ligne de commande ou en service sur la machine à laquelle sont connectés les modules que vous voulez contrôler. Les applications qui veulent se connecter à un virtual hub doivent initialiser l'API en appelant la fonction `yRegisterHub` avec d'adresse IP de la machine faisant tourner le *VirtualHub*, le port par défaut est 4444. Par exemple

```
yRegisterHub('192.168.1.6:4444',errmsg);
```

Si l'application et *VirtualHub* tourne sur la même machine, utiliser l'adresse 127.0.0.1. Consultez la documentation de l'API de programmation [1] pour plus de détails.

4.1. Limitations

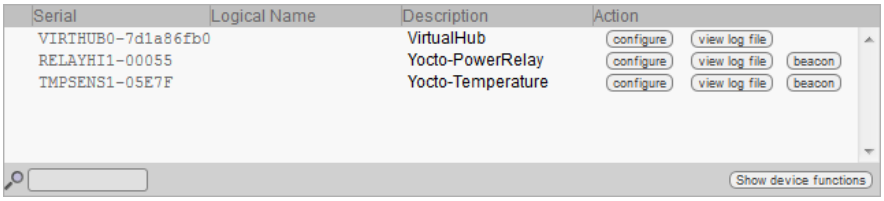
Les modules USB Yoctopuce ont une limitation: sur une machine donnée, il ne peut y avoir qu'une seule application à la fois qui les contrôle nativement. Et il se trouve que le virtual Hub compte pour une application native. En conséquence, si vous tentez de lancer une application qui contrôle nativement des modules Yoctopuce USB, veillez à ce que le *VirtualHub* ne soit pas en train de tourner, que ce soit en ligne de commande ou en service.

Notez que du point de vue programmation, cette limitation peut facilement être contournée en faisant en sorte que vos applications utilisent un *VirtualHub* comme passerelle pour contrôler les modules au lieu de les contrôler directement. Pour ce faire il n'y a qu'un paramètre à changer dans l'appel à `yRegisterHub`.

[1] http://www.yoctopuce.com/FR/libraries

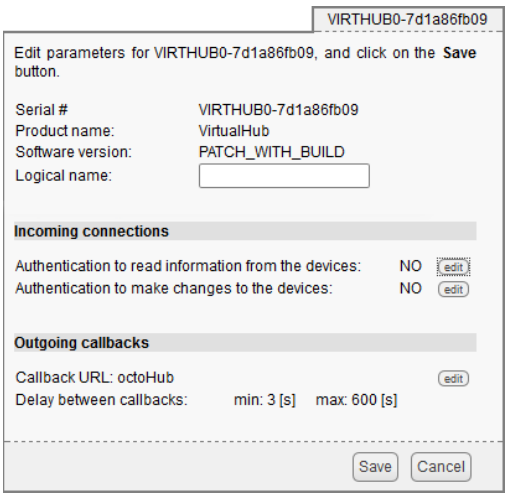
5. Contrôle d'accès

Le *VirtualHub* vous permet d'instaurer un contrôle d'accès à vos modules Yoctopuce. Pour ce faire cliquez simplement sur le bouton **Configure** de la ligne correspondant au virtual hub dans l'interface.



Cliquez sur le bouton "configure" de la première ligne

Cela aura pour effet de faire apparaître la fenêtre de configuration du virtual hub.



La fenêtre de configuration du VirtualHub

Ce contrôle d'accès est contrôlé depuis la section **Incoming connections**. il peut se faire à deux niveaux distincts.

5.1. Accès "admin"

Le mot de passe *admin* verrouille les accès en écriture sur les modules. Lorsqu'il est configuré, seuls les accès de type *admin* permettent d'accéder aux modules en lecture et en écriture. Les utilisateurs utilisant le login *admin* pourront éditer la configuration des modules vus par ce *VirtualHub* comme ils le souhaitent.

5.2. Accès "user"

Le mot de passe *user* verrouille toute utilisation des modules. Lorsqu'il est configuré, toute utilisation sans mot de passe devient impossible. Les accès de type *user* ne permettent d'accéder aux modules qu'en lecture seule c'est à dire pour seulement consulter l'état des modules. Si vous instaurez un contrôle d'accès de type *user*, les utilisateurs utilisant le login *user* ne pourront pas modifier la configuration des modules vus par ce VirtualHub.

Si vous configurez un accès *admin*, sans configurer d'accès *user*, les utilisateurs pourront continuer à consulter vos modules en lecture sans avoir à entrer de mot de passe.

5.3. Influence sur les API

Attention, le contrôle d'accès agira aussi sur les API Yoctopuce qui tenteront de se connecter à ce *VirtualHub*. Dans les API Yoctopuce, la gestion des droits d'accès est réalisée au niveau de l'appel à la fonction `RegisterHub()`: vous devrez donner l'adresse du *VirtualHub* sous la forme *login:password@adresse:port*, par exemple:

```
yRegisterHub("admin:mypass@127.0.0.1:4444",errmsg);
```

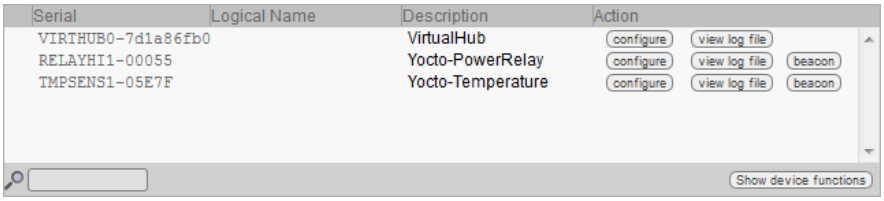
Si vous perdez le mot passe de votre *VirtualHub*, vous pouvez le remettre à zéro en effaçant son fichier de configuration (`.virtualhub.dat`)

6. Interactions avec l'extérieur

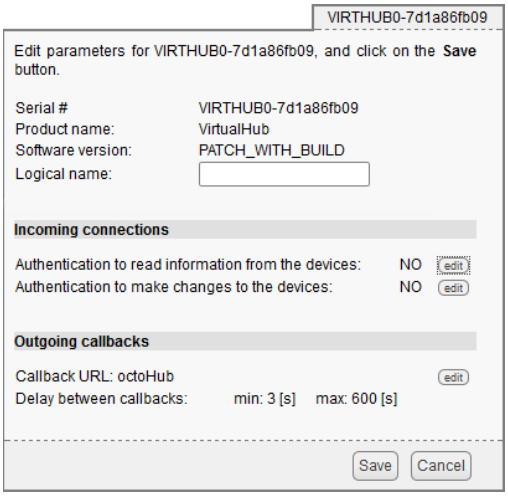
Le *VirtualHub* est capable de poster sur le site web de votre choix l'état des modules qu'il voit. Les valeurs sont postées à intervalles régulier et à chaque fois qu'une valeur change de manière significative. Cette fonctionnalité vous permettra d'interfacer vos modules Yoctopuce avec divers services web.

6.1. Configuration

Pour utiliser cette fonctionnalité, cliquez simplement sur le bouton **Configure** de la ligne correspondant au *VirtualHub* dans l'interface, puis cliquez sur le bouton **edit** de la section **Outgoing calback**.



Cliquez sur le bouton "configure" de la première ligne

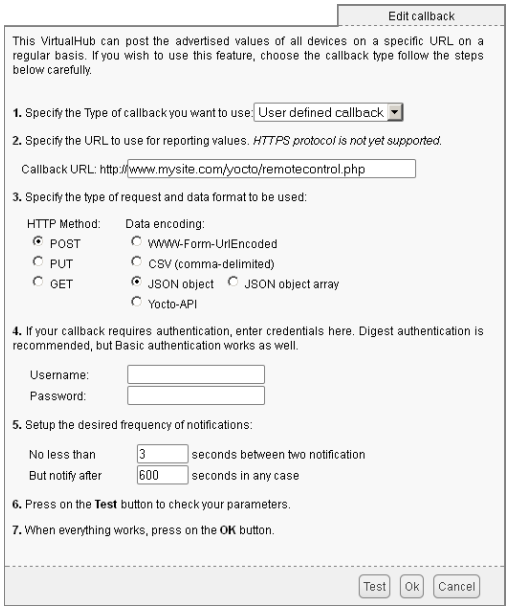


Puis éditez la section Outgoing callbacks.

La fenêtre de configuration des callbacks apparaît. Cette fenêtre va vous permettre de définir comment votre virtual hub va pouvoir interagir avec un serveur WEB externe. Vous avez plusieurs type d'interactions a votre disposition.

6.2. User defined callback

Les "*User defined callback*" vous permettent de personnaliser la manière dont votre virtual hub va interragir avec un site WEB externe. Vous avez besoin de définir l'URL du serveur web sur lequel le *VirtualHub* va poster l'état de ses devices. Notez que seul le protocole HTTP est supporté (pas de HTTPS).


Edit callback
This VirtualHub can post the advertised values of all devices on a specific URL on a regular basis. If you wish to use this feature, choose the callback type follow the steps below carefully.
1. Specify the Type of callback you want to use: User defined callback
2. Specify the URL to use for reporting values. HTTPS protocol is not yet supported.
Callback URL: http:// www.mysite.com/yocto/remotecontrol.php
3. Specify the type of request and data format to be used:
HTTP Method:
POST
PUT
GET
Data encoding:
WWW-Form-UrlEncoded
CSV (comma-delimited)
JSON object
JSON object array
Yocto-API
4. If your callback requires authentication, enter credentials here. Digest authentication is recommended, but Basic authentication works as well.
Username:
Password:
5. Setup the desired frequency of notifications:
No less than 3 seconds between two notification
But notify after 600 seconds in any case
6. Press on the Test button to check your parameters.
7. When everything works, press on the OK button.
Test
Ok
Cancel

La fenêtre de configurations des callbacks

Si vous désirez protéger votre script de callback, vous pouvez configurer un contrôle d'accès HTTP standard sur le serveur Web. Le VirtualHub sait comment gérer les méthodes standard d'identification de HTTP: indiquez simplement le nom d'utilisateur et le mot de passe nécessaires pour accéder à la page. Il est possible d'utiliser la méthode "Basic" aussi bien que la méthode "Digest", mais il est recommandé d'utiliser la méthode "Digest", car elle est basée sur un protocole de question-réponse qui évite la transmission du mot de passe sur le réseau et évite aussi les copies d'autorisation.

Le *VirtualHub* poste avec la méthode POST les valeurs notifiées[1] des modules à intervalle régulier, et à chaque fois qu'une de ces valeurs change de manière significative. Vous pouvez changer les délais d'attente entre les posts.

Tests

Afin de vous permettre de débugger le processus, le *VirtualHub* vous permet de visualiser la réponse au callback envoyé par le serveur web . Cliquez simplement sur le bouton **test** une fois que vous avez renseigné tous les champs. Si le résultat vous paraît satisfaisant, fermez la fenêtre de debug, et cliquez sur **Ok**.

Formats

Les valeurs sont postées sous une des formes suivantes:

1. Si un nom logique a été défini pour une fonction:

```
NOM_LOGIQUE_DE_LA_FONCTION = VALEUR
```

2. Si un nom logique a été défini pour le module, mais pas pour la fonction:

```
NOM_DU_MODULE#NOM_HARDWARE = VALUE
```

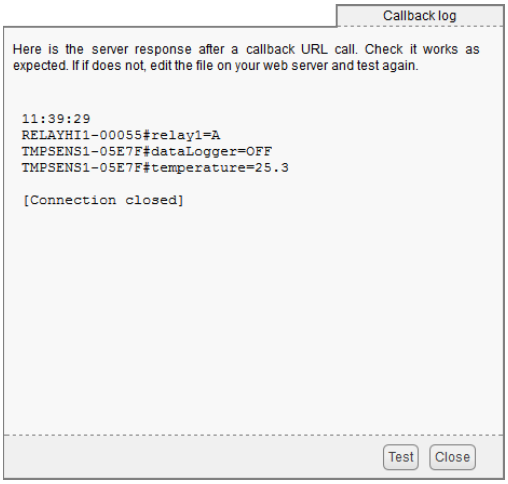
[1] Les valeurs notifiées sont celles que vous voyez quand vous cliquez sur *show functions* dans l'interface principale du VirtualHub

3. Si aucun nom logique n'a été attribué:

```
NUMERO_DE_SERIE#NOM_HARDWARE = VALEUR
```

Voici un script PHP qui vous permettra de visualiser le contenu des données postées par le callback, suivi du résultat dans la fenêtre de debug.

```php
<?php
  Print(Date('H:i:s')."\r\n");
  foreach ($_POST as $key=>$value) {
      Print("$key=$value\r\n");
  }
?>
```


Callback log
Here is the server response after a callback URL call. Check it works as expected. If if does not, edit the file on your web server and test again.
11:39:29
RELAYHI1-00055#relay1=A
TMPSENS1-05E7F#dataLogger=OFF
TMPSENS1-05E7F#temperature=25.3
[Connection closed]
Test
Close

Le résultat du test de callback avec un Yocto-PowerRelay et un Yocto-Temperature.

6.3. Yocto-API callback

L'API Yoctopuce PHP est capable de fonctionner en mode *callback*. Dans ce mode un script PHP peut prendre le contrôle de vos module à travers un filtre NAT sans que vous ayez a ouvrir un port. Typiquement cela permet de contrôler depuis un site WEB publique des modules Yoctopuce installés derrière un router ADSL privé. Le virtual hub sert alors de relais. Vous avez simplement à définir l'URL du script PHP de contrôle et éventuellement les credentials nécessaires pour y accéder. Vous trouvez plus d'informations sur le fonctionnement du mode "callback" de l'API PHP dans le mode d'emploi de vos modules Yoctopuce.

6.4. Cosm

Cosm[2] est un service de cloud gratuit qui vous permettra tracer des graphes avec les données issues de vos capteurs Yoctopuce, et ce sans écrire la moindre ligne de code. Vous avez besoin d'un compte Cosm et de définir un feed et une clef API sur le site de Cosm. Entrez ces deux paramètres dans l'interface du Virtual Hub, c'est tout. Au besoin vous trouverez des explication plus détaillées sur le blog de Yoctopuce[3]. Yoctopuce n'est en aucune manière affilié à Cosm.

[2] www.cosm.com
[3] http://www.yoctopuce.com/FR/article/enregistrer-des-mesures-sur-internet

7. Paramètres de la ligne de commande

Le *VirtualHub* accepte divers paramètres sur la ligne de commande.

-h : aide

Force le *VirtualHub* à afficher une aide succinte

-c : Fichier de configuration

Par défaut le *VirtualHub* stocke son fichier de configuration dans AppData sous Windows et dans le Home directory de l'utlisateur sous Linux et Mac OS X. Cette option permet de changer cet emplacement. Par exemple

```
>virtualhub -c C:\tmp\mysetting.bin
```

-p : changement de port

Par défaut le *VirtualHub* utilise le port TCP 4444, cette option permet d'en utiliser un autre. Par exemple:

```
>virtualhub -p 8889
```

-v : version

Permet d'afficher la version du *VirtualHub*, Par exemple:

```
>virtualhub -v

Version v1.0 (4237)
```

-i : installation du service

Sous Windows, le *VirtualHub* peut fonctionner en service, cette option installe le service et le démarre. Ainsi le *VirtualHub* sera disponible en permance, même si la machine redémarre.

-u : désinstallation du service

Désinstalle le service préalablement installé avec l'option -i (Windows uniquement)

-d : démarrage en service/démon

Sous Linux démarre le *VirtualHub* en arrière plan.

-f : mise à jour du firmware

Mets à jour le firmware d'un module Yoctopuce, pour ce faire vous avez besoin de connaîte le numéro de série du module, et d'un fichier `.byn`. Ces fichiers sont disponibles sur les pages produit du site de Yoctopuce. Exemple de ligne de commande:

```
>virtualhub -f Numero-de-série Fichier.byn
```

-o : Activation de la fonction osControl

Ajoute la fonctionnalité *osControl* au VirtualHub, ce qui permet entre autres d'éteindre à distance la machine qui fait tourner la virtualHub en utilisant l'API yoctopuce.

-A : mise à jour du firmware automatique

Met à jour le firmware de tous les modules Yoctopuce compatibles connectés avec le ficher de firmware donné en paramètre. Ces fichiers de firmware sont disponibles sur les pages produit du site de Yoctopuce. Exemple de ligne de commande:

```
>virtualhub -A Fichier.byn
```

Index

A
Accès 11, 12
Admin 11

C
Callback 14, 15
Commande 17
Configuration 5, 6
Contrôle 11
Cosm 15

D
Defined 14

E
Extérieur 13

F
Firmwares 7

I
Influence 12
Installation 3
Interactions 13
Introduction 1

L
Limitations 9
Linux 3
Localisation 6

M
Modules 5, 6

P
Paramètres 17
Passerelle 9

T
Test 5, 6

U
Upgrades 7
User 12, 14

V
VirtualHub 9

Y
Yocto-API 15